Journal of Korea Multimedia Society Vol. 14, No. 2, February 2011(pp. 153-170)

OMTP BONDI 기반 모바일 웹 위젯 리소스의 효율적 운용 및 구동 성능 개선 기법 연구

방지웅[†], 김대원[††]

요 약

OMTP (Open Mobile Terminal Platform)는 사용자 지향의 모바일 서비스와 데이터 비즈니스의 성장을 목적으로 이동통신 사업자에 의해서 만들어진 국제 포럼이다. BONDI는 OMTP에서 만든 브라우저 기반의 애플리케이션 혹은 위젯이 무선 단말기의 기능을 보안적인 방법으로 접근하게 하는 모바일 웹 런타임 플랫폼이다. 이는 HTML, JavaScript, CSS, AJAX등 웹 표준 기술로 작성된 애플리케이션이 이동 단말기 내부기능에 접근 할 수 있음을 의미한다. BONDI는 웹 run-time 환경에서 단순 네트워크 애플리케이션의 한계를 벗어나 단말 내부의 리소스를 표준안을 통해 접근하게 하기 때문에 OS, 플랫폼에 상관없는 애플리케이션 및 위젯 개발이 가능하다. 웹 브라우저에서 실행되는 위젯은 네트워크 환경에 영향을 받을 수 있으며, 위젯 및 애플리케이션이 무거워 질수록 위젯의 실행속도가 느려질 수 있다는 단점이 있지만 단말기 내부의 네이티브 애플리케이션에 비해 빠르게 웹 리소스를 사용할 수 있고, 사용자가 접근하기 간편한 인터페이스 때문에 꾸준히 사용될 전망이다. 본 논문에서는 OMTP BONDI 웹 위젯 리소스를 효율적으로 운용하고 관리하기 위한 기법을 제안하고 구동 성능 평가 실험을 통하여 개선 결과를 제시 하였다. 실험은 BONDI 위젯 구동 시 사용을 위한 해당 모듈만 로드할 수 있도록 하여 위젯 엔진에서 실행 시 로드되는 모듈 로딩속도를 향상시켜 전체 동작 시간을 개선하고자 하는 목적으로 진행되었다. 이를 위해 BONDI 위젯의 실행속도를 빠르게 할 수 있는 Widget Resource List를 재정의 하고 Widget Cache를 사용하였으며 기존 사용 위젯을 삭제한 후에도 관리할 수 있는 Widget Box를 고안하여 일시적으로 사용하지 않는 위젯을 보관 할 수 있도록 하였다.

An Effective Employment and Execution Performance Improvement Method of Mobile Web Widget Resources Based on the OMTP BONDI

Jiwoong Bang[†], Daewon Kim[††]

ABSTRACT

OMTP (Open Mobile Terminal Platform) is a global forum made by telecommunications providers to promote user-oriented mobile services and data business. Devised by OMTP, BONDI is a browser-based application or a mobile web run-time platform to help widgets make good use of functions of mobile devices in a secure way. BONDI enables applications programmed with web standard technologies such as HTML, JavaScript, CSS, and AJAX to reach the internal functions of mobile devices. Since BONDI, which is not just a simple network application, can reach the internal resources of devices in standard ways, it enables the application and widgets to be developed regardless of the OS or platform.

※ 교신저자(Corresponding Author) : 김대원, 주소 : 충청남도 천안시 안서동 산 29 단국대학교천안캠퍼스 제3과학관 418호(330-714), 전화 : 070)7154-3487, FAX : 070)-7154-3487, E-mail : dr_dwkim@dankook.ac.kr
접수일 : 2010년 8월 18일, 수정일 : 2010년 12월 16일
완료일 : 2011년 1월 8일
[†] 정회원, 단국대학교 대학원 컴퓨터학과
(E-mail : muse@dankook.ac.kr)
[††] 정회원, 단국대학교 공학대학 멀티미디어공학과

Web browser-based widgets are vulnerable to the network environment, and their execution speed can be slowed as the operations of the widgets or applications become heavy. However, those web widgets will be continuously used thanks to the user-friendly simple interface and the faster speed in using web resources more than the native widgets inside the device. This study suggested a method to effectively operate and manage the resource of OMTP BONDI web widget and then provided an improved result based on a running performance evaluation experiment. The experiment was carried to improve the entire operating time by enhancing the module-loading speed. In this regard, only indispensable modules were allowed to be loaded while the BONDI widget was underway. For the purpose, the widget resource list, able to make the operating speed of the BONDI widget faster, was redefined while a widget cache was employed. In addition, the widget box, a management tool for removed widgets, was devised to store temporarily idle widgets.

Key words: Mobile(모바일), OMTP, BONDI, Web(웹), Widget(위젯), Resources(리소스)

1. 서 론

현 시대는 컴퓨터 사용이 대중화 되고, 초고속 통신망의 발달로 많은 사람들이 인터넷과 휴대전화를 시공간의 제약 없이 자유로이 사용하고 있다. 과거 휴대전화는 음성서비스와 SMS 기능만을 고려한 '일반폰'이었다. 3G로의 기술 향상으로 칩, 안테나, 배터리 등 고성능 부품을 채용하게 되어 휴대전화의 성능이 향상되면서 모바일 OS가 탑재되었는데 이로 인해 휴대전화는 점차 단순한 음성서비스, SMS를 넘어 다양한 기능을 사용 할 수 있는 복합적 모바일 디바이스가 되었다. 최근에는 애플의 iPhone과 구글의 Android를 이용하는 콘텐츠를 설치하고 삭제 및 업데이트가 가능한 스마트폰이 등장하였다. 스마트폰의 콘텐츠 중 웹 브라우저를 통하지 않고 날씨, 뉴스, 주식 등의 정보를 바로 이용할 수 있도록 만든 미니 응용프로그램인 위젯과 웹 애플리케이션을 사용하는 사용자 및 개발자가 폭발적으로 늘어나게 되었다. 초기 모바일 위젯은 서비스나 기술적인 접근보다는 휴대전화 내에서 노출이 가장 많은 대기화면 상에서 제공 되었다[1]. 대표적인 서비스로는 KT의 '멀티팝업'과 SKT의 '인터랙티브'등이 있다. 대기화면 상의 위젯은 WAP 브라우저의 접속 없이 휴대폰의 대기화면에 날짜, 뉴스와 같은 컨텐츠 정보를 이용할 수 있게 하여 구동시간을 단축하도록 하였다[2]. 하지만 최근 스마트폰과 Web 2.0의 등장으로 모바일 단말기에서 웹을 활용하려는 많은 연구가 진행중이며, 위젯을 대기화면에서 정보를 제공하는 수단으로 사용할 뿐만 아니라 단말기에 설치되어 동작하는 웹 애플리케이션의 개념으로 확대 사용하고 있다. 웹 애플리케이션이란 HTTP를 통해 전달되는 웹 페이지의 집합체들이 웹 브라우저 내에서 애플리케이션 같은 환경을 제공하는 것으로서 Device API, HTML5와 같은 표준화 작업이 현재 진행 중이다. 특히 W3C MWBP Working Group에서는 모바일 웹을 위한 기술적 모범사례를 표준화 하고 있다[3,4]. 이러한 변화는 스마트폰 시장 활성화와 모바일 플랫폼의 개방화에 따라 위젯이 단말의 홈 스크린 안에 웹 애플리케이션, 웹 위젯 등 다양한 형태로 구분 없이 배치되게 하고 단순한 실행 뿐 만 아니라 정보와 애플리케이션의 업데이트 여부 등을 지속적으로 사용자에게 전달하도록 하고 있다. 하지만 스마트폰마다 다른 형식의 플랫폼을 사용하는 문제와 단말 제조사나 통신사업자 측에서 제공하는 다른 형태의 웹 애플리케이션을 사용해야 하는 단점이 있는 관계로, 이와 무관하게 다양한 플랫폼에서 웹 위젯 등이 동작 가능하도록 OMTP BONDI, JIL(Joint Inovation Lab), Phone Gap 등의 단체에서 모바일 웹 애플리케이션 개발을 위한 연구 및 하이브리드 애플리케이션에 대한 표준 작업을 진행하고 있다[5-7]. 스마트폰에서 다양한 웹 컨텐츠를 활용한 웹 위젯 및 웹 애플리케이션을 실행할 경우 기존 시스템의 한정된 단말 메모리에 비해 상대적으로 많은 메모리를 사용하고 실행 속도가 느린 웹 브라우저 엔진의 사용으로 인해 웹 위젯 및 웹 애플리케이션 실행 시 많은 시간이 소요되고 있다 [8]. 본 논문에서는 이러한 문제점을 해결하기 위해 OMTP BONDI에서 규정한 실행 절차를 간소화하여 실행 시간을 단축하고 그 기능을 향상 시키고자 하였다. 또한 위젯 캐시를 사용하여 이전에 실행했던 위젯들에 대한 실행 시간을 단축하였고 자주 사용하지 않는 위젯들에 대한 관리를 위해 위젯 박스를 제안하여 한정된 단말의 메모리 공간을

효율적으로 활용할 수 있도록 하였다. 또한 본 논문에서 제안한 방법들에 대한 성능 평가를 위해 OMTP BONDI 의 규격에 맞는 시뮬레이터를 구현하여 실험을 진행하였고 성능 평가 후 실행 속도 면에서 기존 시스템 보다 시간이 단축되어 전체적으로 성능이 개선되는 결과를 얻었다.

2. OMTP BONDI

모바일 2.0 에서 가장 주목할 부분은 웹 애플리케이션 기술 분야이다. 웹 2.0의 등장으로 모바일 단말기에서 웹 위젯 애플리케이션을 효과적으로 사용하기 위한 컨텐츠 거래방식, 브라우징 방식 등의 변화가 이루어졌다. 모바일 웹 애플리케이션의 방식은 네이티브와 웹 그리고 하이브리드 애플리케이션으로 나뉜다[9-11]. 표 1에서 네이티브 애플리케이션, 웹 애플리케이션, 그리고 하이브리드 애플리케이션의 특징을 비교하고 있다.

네이티브 어플리케이션은 빠른 속도로 주소록, MMS, 계산기와 같은 모바일 단말기 기능을 효과적으로 사용할 수 있다. 하지만 단말기를 통합적으로 네이티브 어플리케이션을 별도로 개발해야 되기 때문에 어플리케이션의 재활용 및 업그레이드가 용이하지 않다는 단점을 가지고 있다. 반면 웹 어플리케이션은 별도의 설치 없이 업그레이드 된 기능을 사용할 수 있고 Open API를 통해 매쉬업 하는 등 재활용이 가능하다. 하지만 오프라인에서 사용이 불가능하고 브라우저 성능에 쉽게 영향을 받는 단점이 있다. 네이티브 어플리케이션과 웹 어플리케이션의 장점만을 가지는 하이브리드 어플리케이션의 개념이 등장하였다[12]. 최근 애플 iPhone, 구글 Android, 팜 WebOS 등에서는 하이브리드 애플리케이션을 개발할 수 있도록 OMTP BONDI와 JIL등의 모바일 웹 플랫폼 표준을 따르는 웹 런타임(run-time) 엔진을 개발 중이며, W3C에서는 2008년 12월, Device API 관련 워크숍에서 발의하여 2009년 6월 DAP WG을 발족 시켰다[13]. 이로 인해 OMTP BONDI와 JIL의 표준화 내용이 W3C로 취합 되어 웹 애플리케이션 및 위젯의 표준화가 진행 되고 있으며 BONDI는 OMTP 활동 중의 하나로서 브라우저 기반의 애플리케이션 또는 위젯이 모바일 폰 기능에 보안적인 방법으로 접근 가능하게 하는 모바일 웹 런타임 플랫폼이다. 그리하여 2009년 5월 26일 최초 버전인 1.01이 발표 되었으며 W3C 표준 위젯을 사용하고 있다[14,15].

이 논문에서 사용한 BONDI Ver. 1.1 버전의 구조는 그림 1과 같다. 그림 1에서 위젯은 BONDI를 지원하는 대화형 웹 애플리케이션 또는 웹 위젯이다. 웹 위젯은 원격지의 Widget Resource Provisioning Server로부터 위젯 리소스를 모바일 장치 또는 컴퓨터에 다운받아 사용하고, Widget User Agent에 의해 설치, 삭제, 실행, 종료가 가능하다. 위젯은 실행 이전에는 위젯 리소스로 서버 혹은 단말기 내부에 존재하며 리소스는 패키지형식의 압축 파일로 되어 있어 내부에는 HTML, SVG, JavaScript, CSS와 같은 웹 문서와 Configuration Document, Digital

표 1. Native App, Web App, Hybrid App 비교

구 분	Native App.	Web App.	Hybrid App.
Graphic Performance	상	하	상
AppStore 판매	가능	불가능	가능
Offline Mode	가능	일부 가능	가능
웹 서비스 매시업	불가능	가능	가능
Multi-platform 지원	어려움	용이	중간
Storage	로컬	서버 클라우드	모두
Device Capability 이용	용이	불가능	용이
다중 사용자 공동 작업	불가능	가능	가능
소프트웨어 갱신 방법	재설치	사용중 수정	부분 재설치
애플리케이션 재활용성	소스/Lip 활용	소스 및 SaaS	모두
UI 제작 난이도	상	하	중
UI 표현 능력	상	하	중

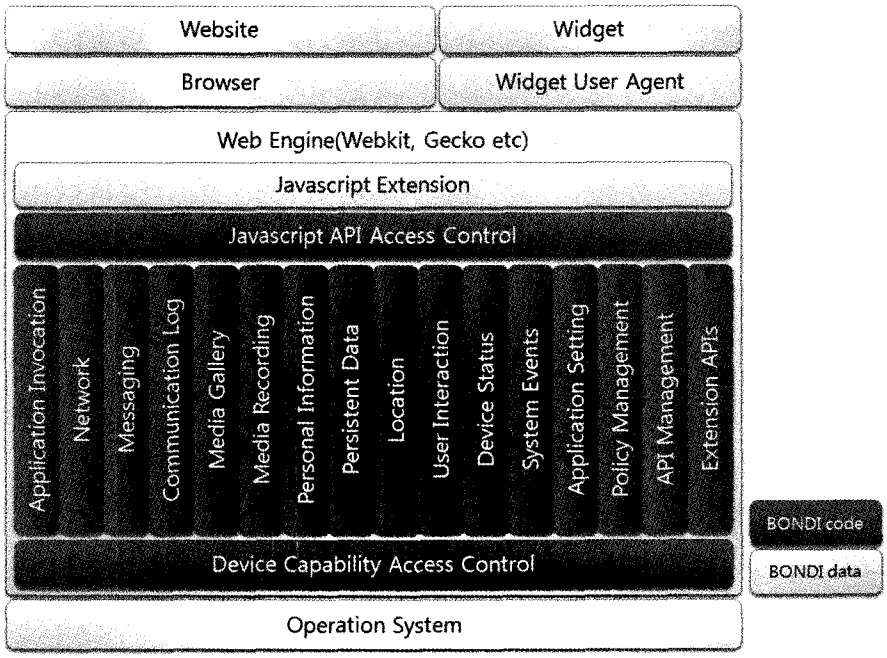


그림 1. OMTP BONDI Ver. 1.1 구조

Signature 등의 정보가 들어있다. Web-site는 HTML, SVG, JavaScript, CSS와 같은 웹 기반 언어로 작성되는데 웹 서버에 정보를 저장하여 인터넷에서 사용자들이 정보가 필요할 때 언제든지 제공될 수 있도록 되어 있다. 웹 애플리케이션은 응용프로그램과 같이 네트워크에 액세스 하며 웹 엔진은 웹 브라우저와 Widget User Agent의 핵심이라 할 수 있다. 위젯 엔진은 기본적으로 JavaScript 엔진을 포함하기 때문에 위젯 리소스의 JavaScript와 CSS 및 HTML, SVG와 같은 마크업 문서의 렌더링을 제공한다. 일반적인 웹 엔진의 종류로는 Web-kit, Gecko 등이 있으며 BONDI 웹 엔진에는 JavaScript Extension과 JavaScript API Access Control, 네이티브 애플리케이션을 지원하고 Extension API가 제공되어 확장이 가능한 BONDI 모듈이 포함되어 있다. Browser는 웹 사이트 컨텐츠를 처리하는 웹 엔진을 사용하는 응용프로그램으로서 일반적으로 북마크 관리 및 탐색기록 등을 사용하여 브라우저 및 웹 엔진 설정을 조정하고 사용자 기능을 제공한다. Widget User Agent는 위젯 리소스를 관리하며 위젯 엔진 위에 설치된 위젯의 설치, 삭제, 실행, 종료 등의 기능을 제공하고 Widget Download Application과 위젯 매니저의 기능을 필요로 한다. Widget Download Application은 하나의 위젯 또는 웹 페이지로도 존재할 수 있으며 위젯 매니저는 위젯의 설치, 삭제, 실행, 종료 등의 기능을 제공한다[16,17]. OMTP BONDI는 구조적으로 웹 브라우저를 기반으로 동작되며 다양한 형태의 모바일 플랫폼 상 동작의 통일성을 위해 독립적인 구동 방식을 지향하고 있다. 또한 런 타임 환경을 구축하여 웹 응용 프로그램이 실시간으로 모바일 플랫폼 환경과 상관없이 용이하게 동작하도록 설계되어 있다. 그러므로 일반적인 네이티브 애플리케이션보다는 웹 브라우저 기반의 웹 애플리케이션이 실행 속도가 느리고 사용하는 메모리 용량 또한 상대적으로 많게 되고 전체적으로 응용 프로그램의 구동 시 발생되는 무형의 비용이 많은 부분을 차지하게 된다[18]. 따라서 OMTP BONDI 위젯의 실행 속도가 느려지고 많은 위젯 메모리 리소스 소비 현상이 발생되어 한정된 용량의 배터리 소모는 물론 전체적인 응용 프로그램의 운용 속도 및 성능 저하를 가져오게 된다. 본 연구에서는 이러한 문제점을 해결하기 위해서 위젯 리소스의 효율적인 운영 및 관리를 위한 위젯 Document 구조 개선, 위젯 캐쉬, 위젯 박스 등을 제안하였고 실험을 통해 성능 개선을 하고자 하였다.

3. 위젯 리소스의 효율적 운용 및 관리 기법

3.1 Widget Configuration Document 요소 추가를 통한 위젯 처리 시간 개선

OMTP BONDI 기반 위젯은 기본적으로 웹 문서를 이용하여 구현된다. 위젯은 HTML text, JavaScript Code, CSS, 이미지 파일 등 실행에 있어 필요한 각종 파일들을 포함하고 있으며 하나의 파일로 압축하여 관리한다. BONDI에서는 위젯 패키지를 zip 파일 형태로 압축하여 사용하며 wgt라는 확장명을 사용한다. 이러한 wgt 파일들을 위젯 리소스라 하며 표 2에서 이를 구성하는 자료들에 대한 정보를 보이고 있다.

표 2의 파일들 이외에 위젯이 필요로 하는 파일들은 추가하여 압축 할 수 있다. 위젯에서 사용하는 이미지나 CSS, JavaScript 등의 파일과 Digital Signatures는 위젯의 성격에 따라 선택적으로 사용할 수 있다. Digital Signatures는 메시지의 진위성과 무결성 및 부인 방지를 보증하기 위한 과정으로서 일반적으로 전자상거래 보안을 위해 사용된다. 위젯 리소스 내에 포함되는 파일 중에 반드시 필요한 파일은 'config.xml'과 'index.htm'이다. 'index.htm'은 Start File로서 위젯 실행 시 가장 먼저 로드되는 문서로서 첫 화면을 구성한다. Start File을 실행하기 전에 User Agent는 위젯에 대한 정보를 알고 있어야 하는데 이러한 정보는 Configuration Document인 'config.xml'에 포함되어 있다. Configuration Document에서는 위젯 정보를 XML의 요소 (Element)와 속성(Attribute)로 정의한다. 표 3은 BONDI Widget Gallery에 있는 한 예로 시계 위젯의 'config.xml' 파일 내용이다.

표 3에서 상위 요소인 <widget>이라는 요소에는 위젯의 형태를 나타내는 xmlns, 위젯 식별 값인 id, 버전 정보를 나타내는 version, 위젯의 높이와 넓이를 나타내는 height, width, 실행 모드를 나타내는 mode 등의 속성 값이 정의 되어 있다. 다음 요소인 <name>은 위젯의 이름을 나타내며, <description>에는 위젯에 대한 설명이 담겨 있다. <author>에는 제작자 이름 및 e-mail에 대한 정보가 있고 <content>에서 src는 위젯의 Start File 경로를 나타내며 <icon>의 src는 아이콘 경로를 나타낸다. <feature>에는 위젯이 사용하는 모듈 이름을 뜻하는 name과 그 모듈에의 접근 가능 여부를 판단하는 required가 있다. 마지막으로 <license>는 위젯에 대

표 2. 위젯 리소스를 구성하는 파일들에 대한 정보

항 목	파 일 명	설 명
Configuration Document	config.xml	위젯에 대한 정보를 포함하고 있는 파일 (위젯 이름, 저자, 라이센스 등)
Digital Signature	signature.xml	위젯 보안을 위한 디지털 서명 정보를 포함하고 있는 파일
Start File	index.htm(html)	위젯 시작 웹 문서이며 실행 시 가장 먼저 로드되는 자료
Widget Icon	icon.png	위젯 아이콘

표 3. 시계 위젯에서 사용하는 'config.xml' 예시

```
<?xml version="1.0" encoding="utf-8" ?>
<widget xmlns="http://www.w3.org/ns/widgets" id="http://bondi.omtp.org/widgets/clock" version="1.02" height="92" width="100" mode="window">
  <name>clock</name>
  <description>Sample widget which tests the UserInteraction interface, including menus, activation/deactivation, screen orientation etc.</description>
  <author email="toby.ealden@gmail.com">OMTP</author>
  <content src="index.htm" />
  <icon src="images\icon.png" />
  <feature name="http://bondi.omtp.org/api.ui" required="true"/>
  <license>Licensed to OMTP Ltd. (OMTP) under one or more contributor license agreements. See the NOTICE ..... You may obtain a copy of the License at http://bondi.omtp.org/BONDI</license>
</widget>
```

표 4. 〈feature〉에서 사용할 수 있는 API 권한 목록

BONDI 모듈	모듈 URL	설 명
applauncher	http://bondi.omtp.org/api.applauncher.launch	위젯 및 내부 애플리케이션에 대한 실행 API 권한
messaging	http://bondi.omtp.org/api..messaging.email.attach 외 5개	단말기에서 SMS, MMS, E-Mail 자원에 대한 API 사용 권한
ui	http://bondi.omtp.org/api.ui	그래픽 자원 API 사용 권한
filesystem (gallery)	http://bondi.omtp.org/api.io.file.read 외 1개	File 입출력 API 사용 권한
devicetatus	http://bondi.omtp.org/api.devicestatus.get 외 2개	단말기 자원에 대한 접근 API 사용 권한
appconfig	http://bondi.omtp.org/api.appconfig.get 외 1개	위젯 및 애플리케이션의 환경 설정 API 사용 권한
geolocation	http://bondi.omtp.org/api.location.position	GPS 장치 API 사용 권한
camera	http://bondi.omtp.org/api.camera.get 외 4개	카메라 관련 API 사용 권한
commlog	http://bondi.omtp.org/api.commlog.sms.get 외 3개	SMS, MMS, E-mail, Call Log 데이터 API 사용 권한
PIM	http://bondi.omtp.org/api.pim.contact.read 외 5개	PIM 관련 API 사용 권한

한 라이센스 정보를 수록하고 있다. Configuration Document 요소 중에 <feature>는 위젯이 사용하고자 하는 모듈에 대한 정보를 포함하고 있는데 이는 단말에서 사용 가능한 사운드 장치, 디스플레이 장치, 파일 시스템, 메모리 등을 이용하도록 제공되는 API를 말한다. 따라서 위젯 내부에서 정의되는 BONDI API를 사용하기 위해서는 <feature>에 사용 하고자하는 API의 권한을 요구하는 정보를 포함해야 한다. 또한 모듈 이름은 인터넷 상의 파일 주소인 URL로 표기한다. 표 4는 <feature>에 사용할 수 있는 API 권한 정보를 모듈 별로 나타낸 것이다.

이러한 권한 정보는 실행 시 위젯 엔진이 읽고 내부에서 요청하는 모듈에 해당하는 API에만 접근하도록 규정한다. 이는 단지 접근 권한이 없는 API에 대해서는 사용하지 못하도록 막는 역할을 수행하기도 한다. 위젯은 위젯 엔진에 의해 실행되는데 이때 위젯 엔진은 기본적으로 BONDI에서 제공하는 모든 모듈을 로드하게 된다. 그리고 실행되고 있는 위젯이 API 접근을 요청하면 <feature> 요소에 기록되어 있는 API 권한을 확인하고 해당 모듈 API에만 접근을 허용한다. BONDI의 이러한 점은 부가적인 메모리 요구와 불필요한 모듈 로드로 인한 위젯 처리 시간을 증가시키며 불합리한 특성을 지니고 있다. 이에 본 논문에서는 위젯 구동 시 효용성 향상을 위해 실행되는 위젯이 사용하지 않는 모듈에 대해서는 위젯 엔진에서 로드하지 않고 실제 요구 되어지는 모듈만을 선택적으로 로드하는 방법을 연구하였다. 이를 위해 Configuration Document의 <feature> 요소에 새로운 속성 값인 'loaded'를 추가하였다. 표 5는 기존 <feature> 요소의 단일 선언 방법에 대한 예시이다.

표 5에서 첫 번째 name 속성은 사용하고자 하는 모듈의 URL을 기록한다. 두 번째 required 속성은 API 접근을 허용하겠다는 의미로는 'true', 사용하지 않겠다는 의미로는 'false'를 기록하고 있다. 이의 운용 향상을 위해 위젯 엔진에서 해당 모듈만을 로드하도록 하는 'loaded'라는 속성을 추가한다. 'loaded'는 해당 모듈만을 로드하겠다는 의미로는 'true', 로드하지 않겠다는 의미로는 'false'를 사용하도록 한다.

표 5. 〈feature〉 요소의 단일 선언 방법 예시

```
<feature name="http://bondi.omtp.org/api.ui" required="true" />
```

표 6. 〈feature〉 요소에 loaded 속성이 추가된 선언 방법 예시

```
<feature name="http://bondi.omtp.org/api.ui" required="true" loaded="true"/>
```

'loaded' 속성 값을 사전에 정의하지 않았다면 기본 값은 'false'로 한다. 표 6은 <feature> 요소에 loaded 속성이 추가된 선언 방법을 보이고 있다.

이처럼 <feature> 요소에 추가된 loaded 속성을 이용하여 위젯 엔진은 실행되는 위젯이 필요로 하는 모듈만을 로드할 수 있게 된다. 따라서 위젯 엔진의 불필요한 메모리 사용량을 줄이고 위젯 처리 시간도 단축할 수 있게 된다. 또한 위젯이 저장된 단말 기기에는 수많은 개인 정보 및 중요한 정보들이 많이 저장되어 있는데 해당 모듈만 기동함으로써 악의적인 개발자에 의해 개발된 위젯으로 인한 개인 정보 누출 위협 및 피해 등의 확률을 현저히 낮출 수 있다.

3.2 Widget Resource List 재 정의를 이용한 위젯 처리 시간 향상

BONDI에서는 설치된 위젯에 대한 정보를 관리하기 위해서 Widget Resource List (WRList)를 사용한다. WRList는 일종의 데이터베이스로서 파일이나 단말 내에 데이터베이스 응용프로그램에 의해 구현, 저장된다. User Agent의 Widget Installer에 의해 위젯이 설치될 때 위젯 매니저는 설치 경로와 시간 정보를 WRList에 기록한다. 표 7은 OMTP BONDI에서 규정하는 WRList의 구조를 나타낸다.

표 7. BONDI에서 규정하는 Widget Resource List 정보

항 목	내 용
Widget URL	위젯 설치 경로
Time stamp	위젯이 설치되는 시간 정보

위젯을 실행하기 위한 정보는 위젯 리소스 (*.wgt) 내 'config.xml'에 기록 되어 있기 때문에 WRList는 위젯에 대한 설치 경로 및 설치 시간 정보만 기록하고 관리한다. 위젯 매니저는 위젯을 실행할 때 엔진에게 WRList 정보 중 Widget URL을 제공하며 엔진은 Widget URL에 있는 리소스 내 'config.xml'을 읽어 위젯 실행에 필요한 정보를 얻는다. 그림 2는 위젯의 실행 과정을 나타내는 신호 흐름도이다.

위 그림에 따르면 열 한 단계를 통하여 사용자가 실행 시키고자 하는 위젯이 화면에 출력 되게 된다. 이러한 많은 단계들로 인해 위젯 처리 시간이 증가되는데 6-9단계가 위젯 실행 시간의 대부분을 차지하고 있다. 그 이유는 위젯 리소스가 zip으로 압축되어 있기 때문이다. 6단계에서는 BONDI 규정에 맞는 위젯인지를 판별하기 위해 위젯 리소스의 압축을 해제한다. 이 과정에서 위젯 리소스 내에 포함되어 있는 자료들이 많을 경우 시간이 그 만큼 더 걸린다고 볼 수 있다. 7단계에서 디지털 서명이 있을 경우 8단

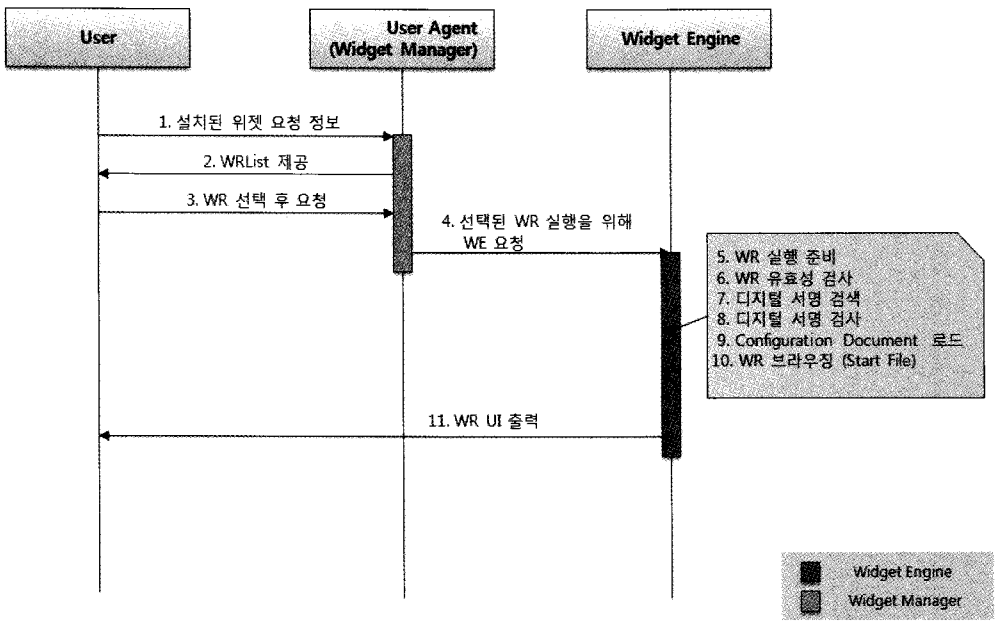


그림 2. BONDI에서 규정하는 위젯의 실행 과정 흐름도

계에서 디지털 서명에 대한 검사를 실시하는데 RSA (Rivest Shamir Adleman)와 같은 암호화와 복호화 과정을 통한 인증 시간이 소요된다. 마지막으로 9단계에서 위젯 실행 정보를 얻기 위해 'config.xml'을 로드하는데 시간이 소요된다. XML로 이루어진 Configuration Document는 XML Parser로 읽어 들이기 때문에 적지 않은 시간이 소요된다. 이상 서술된 6-9단계에서 위젯 실행 시간이 길어지는 현상이 발생하는데 이러한 문제점을 해결하고 기존 BONDI에서 규정하는 위젯 실행 절차를 간소화하기 위해 본 연구를 통해 새로운 Widget Resource List를 고안 하였다. 9단계에서는 위젯을 실행할 때 필요한 정보가 리소스 내에 있기 때문에 매번 'config.xml' 파일을 로드해야 된다. 이 'config.xml'에는 위젯 실행에 필요한 정보들이 기록되어 있고 이러한 정보를 위젯 설치 시에 WRList에 기록하면 9단계에서 수행해야하는 XML Parsing 과정을 생략하여 전체 실행 시간을 줄일 수 있다. 표 8은 이러한 위젯 정보를 기록하기 위해 재 정의된 WRList의 구조를 나타낸다.

표 8에서 WRList에 새로이 추가된 정보로는 위젯 식별 값, 이름, 설명, 버전, 출력 모드, 출력 넓이, 높이, Start File 경로, 아이콘 경로 등이 있다.

또한 본 논문의 3.1절에 서술된 'Configuration Document에 loaded 요소 추가를 통한 위젯 처리 시간 개선' 방법에서 필요한 권한 정보들을 Module-Permissions에 기록하여 위젯 엔진 실행 시 신속히 적용하도록 하였다. 그림 3에서 재 정의된 WRList의 위젯 정보를 추가하기 위해서 위젯 설치 절차를 변경한 신호 흐름도를 보이고 있다. BONDI에서 규정하는 위젯 설치 절차에 9-12단계를 추가하였다. 절차 추가로 인한 설치 시간이 기존의 방법보다 더 소요되지만 위젯 구동 시 실제 걸리는 시간을 줄일 수 있다는 이점이 있다. 다시 말해서 위젯 리소스 설치는 한 번 뿐이지만 실행 절차는 사용자의 요청이 있을 때마다 발생하기 때문에 이를 절약하는 효율적인 위젯 운용이라고 볼 수 있다. 또한 10단계에서는 보안 경고를 통해 사용자로 하여금 설치되는 위젯이 단말의 어떤 자원에 접근하는지에 대한 판단을 할 수 있게 하여 개인 정보 보호 차원에서 안정성을 제공하며 이는 9단계에서 'config.xml' 파일을 로드하기 때문에 가능한 것이라고 볼 수 있다. 이렇게 재 정의된 WRList 정보를 위젯 실행 시에 활용할 수 있도록 BONDI에서 규정하는 위젯 실행 과정을 변경하였고 그림 4에서 해당 신호 흐름도를 보이고 있다. 여기서는 기존 BONDI에서 규정하는 위젯 실행 절차와는 달리 6단계에서 실시했던 유효성 검사는 하지 않는데 이는 위젯 설치 과정에서 리소스의 유효성 검사를 이미 했기 때문이다. 또한 위젯 정보를 'config.xml'로부터 얻기 위하여 XML Parsing 과정을 생략하고 WRList로부터 해당 위젯 정보를 직접 가져오도록 하였다. 이리하여 위젯 실행 시간을 단축시키고 위젯 리소스 설치 과정에 보안 경고 과정을 포함시켜 효율

표 8. 재 정의된 Widget Resource List 구조

순 번	항 목	내 용	비 고
1	Widget ID	위젯 식별 값	새롭게 추가된 정보
2	Widget Name	위젯 이름 혹은 제목	
3	Widget Description	위젯 설명	
4	Widget Version	위젯 버전	
5	Widget Mode	위젯 출력 모드	
6	Widget Width	위젯 출력 넓이	
7	Widget Height	위젯 출력 높이	
8	Widget Content URL	위젯 Start File 경로	
9	Widget Icon URL	위젯 아이콘 경로	
10	API-Permissions	BONDI API 사용 권한	
11	Module-Permissions	BONDI 모듈 사용 권한	
12	Widget URL	위젯 설치 경로	BONDI
13	Time Stamp	위젯이 설치되는 시간 정보	

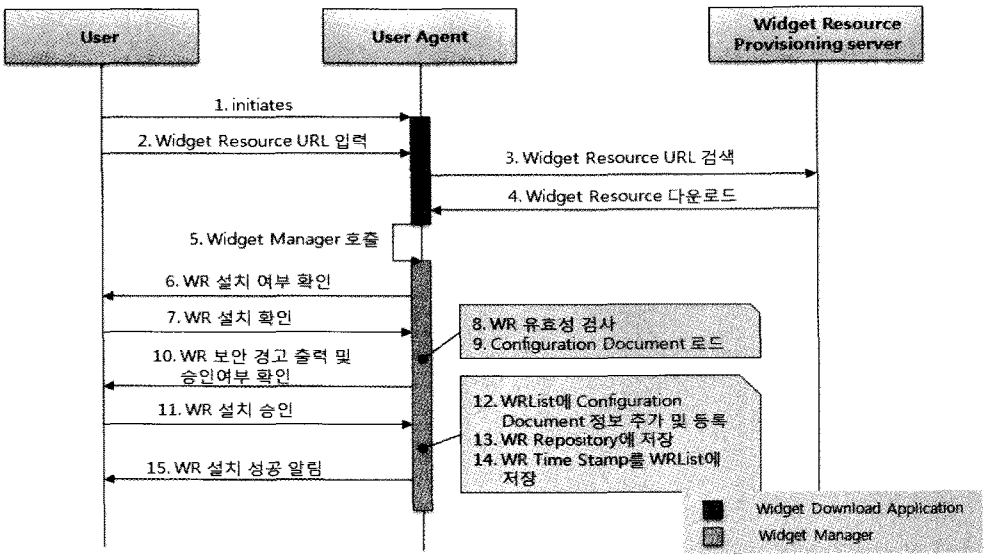


그림 3. 재 정의된 WRList에 위젯 정보 기록을 위한 위젯 설치 과정 흐름도

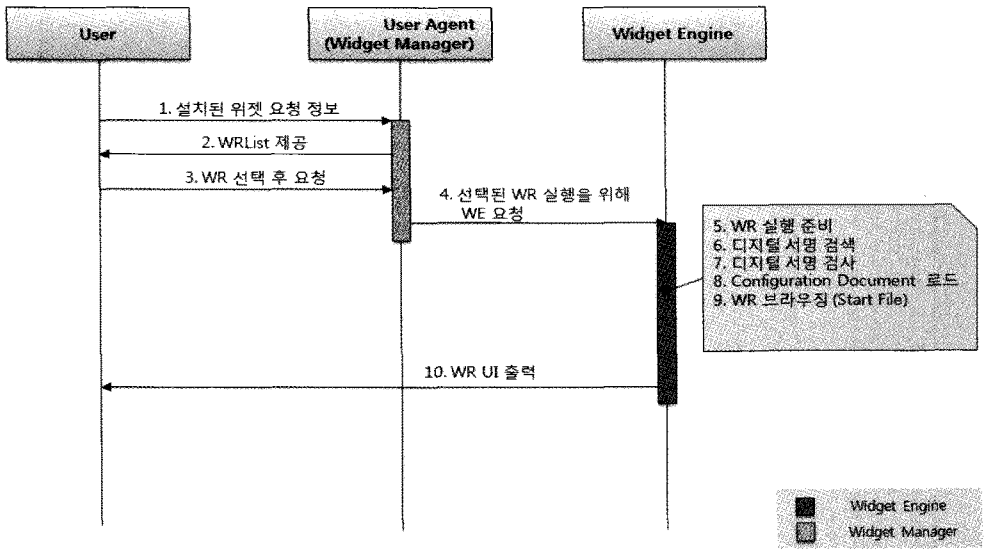


그림 4. 재 정의된 WRList를 이용한 위젯 실행 과정 흐름도

적이고도 신속한 위젯 설치 및 운용 관리를 할 수 있도록 하였다.

3.3 Virtual Widget Cache를 이용한 위젯 실행 시간 개선

위젯 실행 절차에 가장 많은 영향을 주는 요소는 압축 해제이다. 위젯 리소스는 zip으로 압축되어 있어 매번 실행을 위해 압축 해제를 해야 하는데 대용량 파일을 포함하고 있는 경우 해제 시간이 길어진다. 이 시간을 줄이기 위해 본 연구에서는 Virtual Widget Cache를 고안하였으며 일반 컴퓨터에서 사용하는 가상 메모리와 유사하게 주기억 장치가 아닌 보조 기억 장치의 일정 메모리 공간을 주기억 장치 공간으로 사용하였다. BONDI에서는 위젯 리소스 저장소를 SD Card나 SDRAM 내부 폴더에 지정하는데 여기에 임시 폴더를 생성하여 위젯을 실행하기

위한 공간으로 활용한다. 임시 폴더는 위젯이 실행될 때마다 생성 및 삭제를 반복하고 Virtual Widget Cache는 리소스 저장소에 자주 사용하는 위젯 리소스를 실행 시에 압축 해제한 후 삭제하지 않고 유지함으로써 다음에 해당 위젯을 실행 했을 때 구동 시간을 절약할 수 있도록 한다. 또한 Virtual Widget Cache 관리를 위해서 User Agent 내에 Widget Cache Manger를 추가하였는데 이는 자주 사용되는 위젯들의 정보와 Virtual Widget Cache 내에 저장되어 있는 데이터의 관리를 목적으로 한다. Widget Cache Manger는 위젯 정보의 효율적 관리를 위해 Widget Cache Table을 사용하며 표 9에 그 구조를 보이고 있다.

표 9에서 Virtual Widget Cache 내에서 관리되는 위젯의 수를 Widget Number로 나타내고 있는데 이는 관리 가능한 최대 위젯의 수를 지정하고 있으며 Widget Cache Manager를 이용하여 변경할 수 있다.

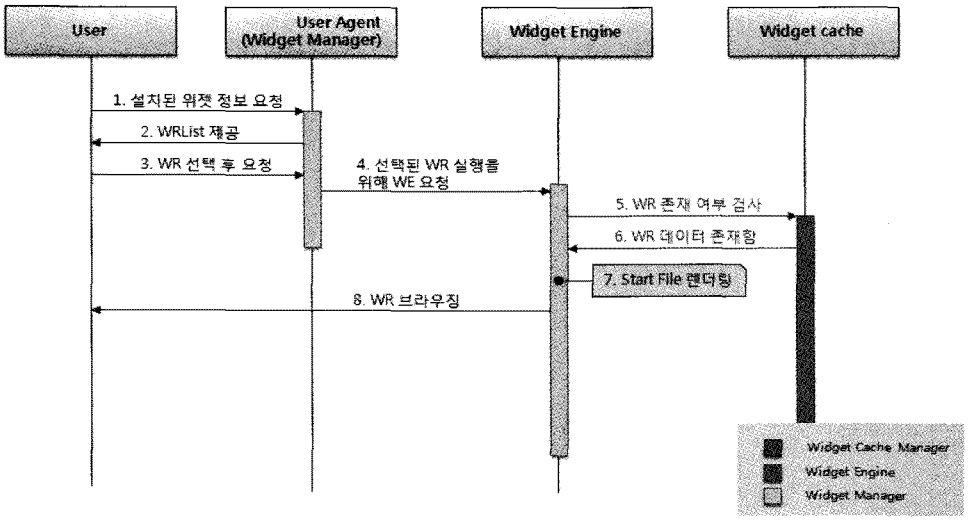
Widget ID는 WRList에 존재하는 위젯의 ID를 나타내며 위젯 실행 시 엔진이 Virtual Widget Cache 내에 실행하고자 하는 위젯 데이터가 존재하는지 여부를 판단할 때 사용된다. Widget Last Execute Time은 위젯이 마지막으로 실행되었던 시간을 나타내며 위젯 종료 시 시간이 기록된다. Widget Frequency Count는 위젯이 실행된 횟수를 뜻하는데 한 개의 위젯을 중복 실행한 경우는 제외한다. Widget Cache Manager는 Widget Last Execute Time과 Widget Frequency Count를 참고하여 Virtual Widget Cache를 관리하게 된다. 예를 들어, Virtual Widget Cache 내에서 저장 가능한 위젯의 최대 수가 5이고 현재 cache 내에 저장되어있는 위젯이 5라고 할 때, 사용자가 새로운 위젯을 실행하게 되면 Widget Cache Manager는 현재 cache에 기록할 수 없음을 확인한 후 cache 사용을 위해 다섯 개의 위젯 중 하나를 삭제한다. 이 때 Widget Frequency Count가 가장 낮은 것을 최우선으로 삭제하고 값이 같은 위젯들이 존재하면 그 중 Widget Last Execute Time이 가장 오래된 위젯을 cache 내에서 삭제한다. 그림 5는 Virtual Widget Cache를 사용하여 위젯을 실행하는 과정을 나타내는 두 종류의 흐름도이다. 이 그림의 (A)는 1-8단계에서 Virtual Widget Cache 내에 실행하려는 위젯 데이터가 존재할 경우 위젯 리소스의 압축 해제 과정 없이 곧바로 Start File을 로드하여 출력하는 과정을 보이고 있다. 또한 디지털 서명의 경우 cache 내에 기록하기 전에 (B)의 9단계에서 디지털 서명을 실시하였으므로 생략해도 무방하다. 위젯 데이터가 Virtual Widget Cache 내에 존재하지 않을 경우에는 그림 5 (B)의 절차 단계를 거치기 때문에 위젯 실행 시간 감소 효과는 얻지 못한다. 하지만 특정 애플리케이션이 빈번하게 사용될 경우 Virtual Widget Cache는 효율적으로 사용될 수 있으며 실행 절차의 축소로 인해 위젯 실행 시간 단축이 가능하다.

3.4 Widget Box를 이용한 위젯 리소스 보관 및 관리

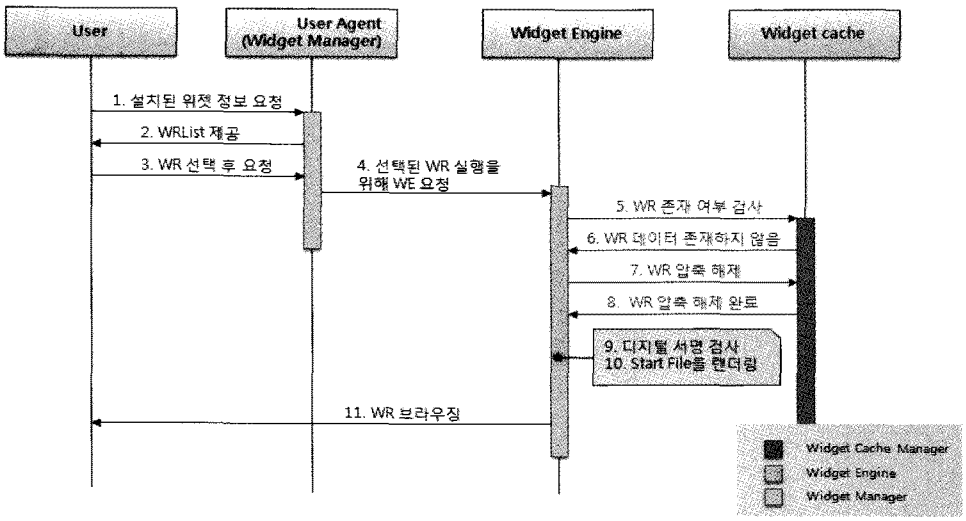
휴대 단말에서 소용량 메모리로 인해 더 이상 위젯을 저장할 수 없는 경우가 있다. 또는 잘 사용하지 않는 위젯을 삭제하거나 지금 당장 필요하지 않아서 사용하지는 않지만 소멸 시키지 않고 보관하고 싶은 위젯이 있을 수 있다. 이 때 설치했던 위젯을 단말에서 제거한 후 원격 서버에 보관할 수 있는 개인화된 공간이 바로 Widget Box이다. 사용자는 단말에서 위젯을 제거하기 전 원격지에 있는 Widget Box에 Widget Uploader를 이용하여 위젯 리소스를 전송하고 단말 상 위젯을 삭제한다. 그림 6은 삭제하려는 위젯 리소스를 Widget Box에 업로드 하는 절차를 나타내는 흐름도이다. Widget Box에 보관된 위젯 리소스는 나중에 사용자가 다시 접속하여 다운로드 받은 후 사용이 가능하다. Widget Box는 Widget Resource Provisioning Server에 위치하고 있으며 사용자가 자신의 Widget Box로부터 보관된 위젯을 다운로드하여 설치하는 과정은 기존 BONDI에서 규

표 9. Widget Cache Table 구조

항 목	설 명	비 고
Widget Number	위젯 번호	
Widget ID	위젯 식별 값	
Widget Last Execute Time	위젯이 마지막으로 실행되었던 시간	
Widget Frequency Count	위젯이 실행 횟수	중복 실행 제외



(A) 위젯 캐시에 위젯 리소스가 존재할 때



(B) 위젯 캐시에 위젯 리소스가 존재하지 않을 때

그림 5. Virtual Widget Cache를 이용한 위젯 실행 과정 흐름도

정하는 내용과 동일하다. Widget Box는 단말 내 다수의 위젯 설치로 인한 메모리 부족 현상을 완화시킬 수 있으며 사용자가 구입한 유료 위젯 리소스를 삭제했을 때 발생하는 금전적 손실을 방지할 수 있는 이점이 있다.

4. 실험 및 성능 평가

4.1 Simulation system

그림 7은 본 논문에서 제시한 위젯 리소스 관리와 운용 방법 및 이와 관련하여 BONDI에서 표준화 한 내용을 실험하고 성능 평가를 하기 위해 구축한 시스템 구조도이다.

그림 7에서 시뮬레이션 시스템은 실제 위젯이 실행되는 'Mobile Client'와 위젯 패키지를 다운 받을 수 있는 'Widget Resource Provisioning Server'로 나누어진다. 'Mobile Client'에는 웹 문서로 제작된 위젯을 실행하는 'Browser'가 있는데 이 내부에는 웹 위젯을 화면에 출력하는 'Widget View'와 HTML 코드를 parsing 처리하는 'HTML Engine', 웹 문서의

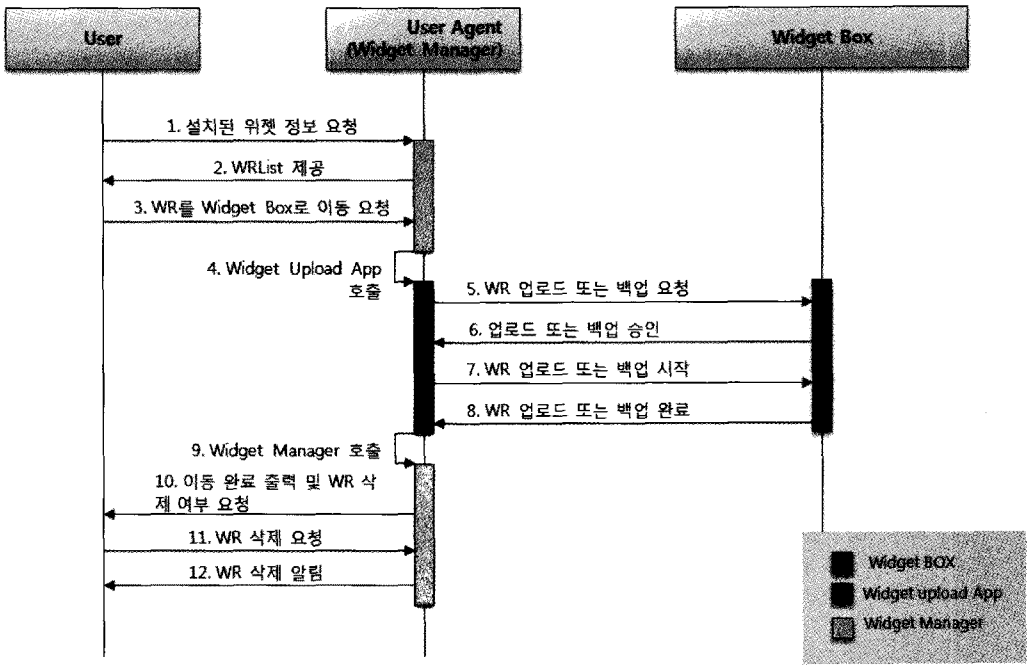


그림 6. Widget Box에 리소스 업로드 및 위젯 삭제 절차 흐름도

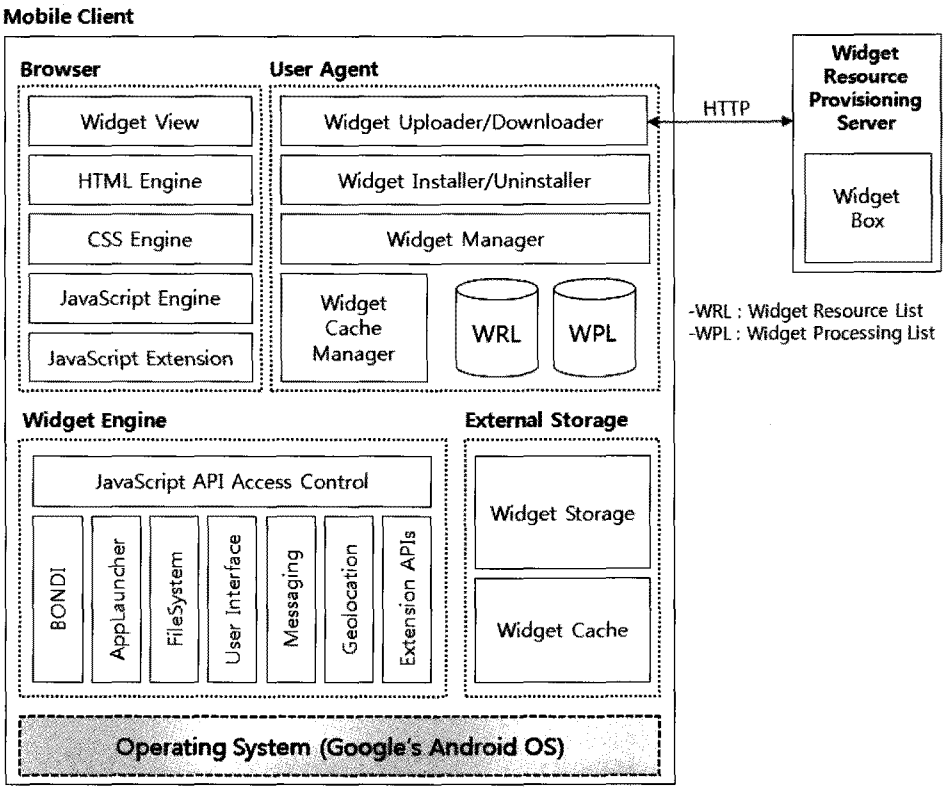


그림 7. Simulation system 구조도

전반적인 스타일을 미리 저장해 둔 스타일 시트를 처리하기 위한 'CSS Engine' 등이 있다. 또한 웹 문서 내에서 사용된 JavaScript 코드 처리를 위한 'JavaScript Engine'이 있으며 BONDI에서 제공하는 위젯 관련 API를 사용하기 위한 'JavaScript Extension'이 존재한다. 'User Agent' 내부에는 원격지의 서버에서 위젯 패키지를 다운 받기 위한 'Widget Downloader'와 Widget Resource Provisioning Server에 존재하는 Widget Box에 위젯을 업로드하기 위한 'Widget Uploader'가 있다. 또한 다운로드 받은 위젯 패키지의 설치와 삭제를 담당하는 'Widget Installer/Uninstaller'가 있으며 'Widget Manager'는 설치된 위젯을 실행하거나, 환경 설정, WRL (Widget Resource List) 또는 WPL (Widget Processing List)에 추가, 삭제, 수정하는 등의 작업을 담당한다. 본 연구에서 제안한 Widget Configuration Document 요소 추가를 통한 위젯 처리 시간 개선 및 Widget Resource List 재 정의를 이용한 위젯 처리 시간 향상을 위해 Widget Manager와 Widget Installer/Uninstaller 부분에 있는 기존의 OMTP BONDI 처리 절차를 본 논문에 제시된 절차로 변경하였다. 'WRL'은 설치된 위젯 목록 관리를 위해 사용되는 데이터베이스이며, 'WPL'은 현재 실행되고 있는 위젯에 대한 정보를 담고 있는 데이터베이스이다. 'Widget Cache Manager'는 본 논문에서 제시한 방법에 대한 성능 평가를 위해 추가 구축된 모듈로서 위젯 실행 시 빈번하게 사용되는 데이터를 'External Storage'에 기록한 후 'Widget Manager'를 이용한 위젯 실행 시 즉시 참조하여 사용할 수 있도록 하는 모듈이다. 'External Storage'는 다운로드 받았거나 이미 설치된 위젯 패키지를 보관하는 외부 메모리 저장소를 뜻하는 'Widget Storage'와 위젯 cache 데이터를 저장하는 'Widget Cache'로 구성되어 있다. 위젯 실행에 있어 가장 중요한 역할을 담당하는 'Widget Engine'은 BONDI에서 제공하는 API를 지원하기 위한 'JavaScript API Access Control'을 상위 레벨에 배치하고 있으며 BONDI API 기능을 위한 모듈인 'BONDI', 'AppLauncher', 'FileSystem', 'User Interface', 'Messaging', 'Geolocation', 'Extension APIs' 등을 하부 구조에 보유하고 있다. 마지막으로 원격지에서 위젯 패키지를 다운로드 하거나 업데이트 할 수 있도록 버전 관리를 담당하는 'Widget Resource Provisioning Server'가 있다. 모바일 클라이언트와 서버는 HTTP를 이용하여 설치된 위젯에 대한 업데이트 및 새로운 유무료 위젯에 관한 정보를 제공하며 Widget Box에 사용자의 위젯을 다운로드 또는 업로드 할 수 있는 기능을 제공한다.

Widget Box는 본 연구에서 새로이 제시한 부분으로서 이를 이용한 위젯 리소스 보관 및 관리 기능이 가능하도록 추가되었다. 성능 평가를 위한 시뮬레이터는 Android 플랫폼 2.0이 탑재된 Motorola 사의 XT720에서 구동 되었으며 S/W 구현을 위해 자바 언어가 사용 되었다. 표 10은 Motorola의 XT720 시스템 사양을 나타낸다. 그림 8은 성능 평가에 사용된 시뮬레이터의 실행화면을 나타낸다.

성능 평가에 쓰일 샘플 BONDI 위젯 패키지 열 개가 선별되었고 시뮬레이션 프로그램을 이용하여 본 논문에서 제시한 방법들에 대한 성능 평가가 이루어졌다. 선별된 열 개의 위젯은 Aplix Web Commander, GPS Monitor, GPS Speedometer, simon 0870, Ajax demo, API Diagnostics, WWMM, Bike Sharing, Send SMS, App Manager 이며 이들은

표 10. Motorola XT720 시스템 사양

항 목	내 용
디스플레이 타입	WVGA, 16M Color
디스플레이 크기	16:9, 480×854
타입	Touch Tablet
기본 메모리	512MB NAND Flash, 256MB SDRAM
CPU	550Mhz
외장 메모리	8GB (SDCARD)
운영체제	안드로이드 2.0

그림 8.Android 2.0 Target Device 및 구동 중인 응용프로그램

OMTP BONDI 웹사이트 내 'Widget Gallery' (http://bondidev.omtp.org/widget-gallery/default.aspx)에서 다운로드 받아 사용되었다. 참고로 'Widget Gallery'의 샘플 위젯들은 상업 목적이 아닌 모듈 개발 및 연구용 위젯들로서 BONDI APIs 1.1에서 지원하는 API들을 테스트 하고 향 후 발전시키기 위해 공개된 위젯들이다.

4.2 성능 평가 및 결과 분석

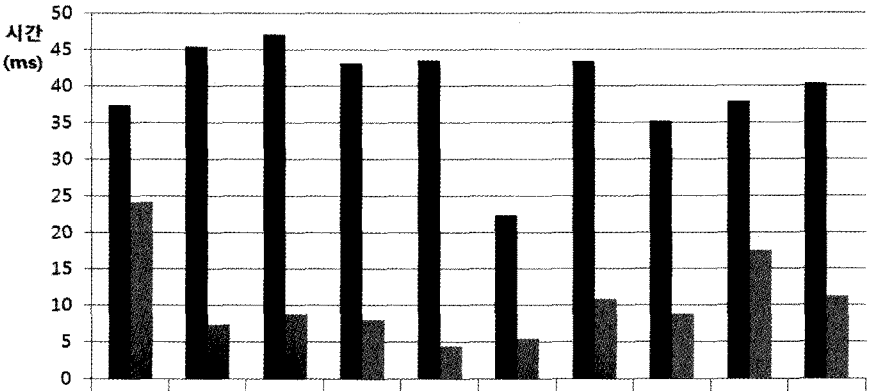
다양한 위젯 종류가 존재하고 각각의 사용 모듈이 다른 관계로 성능 평가 및 측정에 영향을 끼치는 요소를 최소화하기위해 다섯 개의 모듈과 열 개의 위젯을 선택하여 실험 하였다. 실험 결과를 비교하기 위한 대조군으로서 기존 BONDI에서 제안한 방법을 사용하여 위젯 구동 속도를 측정하였으며 실험군으로는 본 연구에서 제시한 방법으로 각각의 위젯 실행 속도를 측정하고 비교 분석 하였다.

4.2.1 Widget Configuration Document 요소 추가를 통한 성능 평가

그림 9는 선별된 열 개의 BONDI 위젯에 'Widget Configuration Document 요소 추가를 통한 실행 시간 개선 기법'을 적용하여 측정한 값을 기존 BONDI에서 규정한 방법을 이용하여 측정한 데이터와 비교한 그래프이다. 그림 9에서 위젯의 메모리 사용량이나 사용 모듈이 달라 약간의 시간 차이는 있지만 각 위젯들의 모듈 로딩 시간이 크게 줄어든 것을 확인할 수 있다. 가장 큰 차이를 보인 위젯은 Ajax demo인데 이는 BONDI 모듈을 사용하지 않는 위젯이기 때문이다. 반면 Aplix Web Commander 위젯은 가장 작은 속도차이를 보였는데 여기서 사용하는 모듈은 UI 모듈과 File-system 모듈로서 그 용량이 크기 때문이다. 그림 10은 위젯이 브라우저를 통해 스크린에 완전히 로딩 될 때까지의 시간을 측정한 결과를 보이고 있다. 여기서는 서로간의 시간 차이가 크게 나타나지 않았으며 추가적인 위젯 실행 속도 개선 기법을 더하여 4.2의 (2)절과 (3)절에서 그 결과를 보이고 있다. 그래프를 분석하면 다른 위젯에 비해 GPS Monitor, simon0870, Ajax Demo, Send SMS 등 위젯의 경우 Full-screen으로 전환되기까지의 시간이 오히려 더 오래 걸리는 것을 알 수 있다. 이는 위젯의 리소스(*.wgt) 용량이 크고 이미지, CSS, JavaScript 등의 파일이 많기 때문이다.

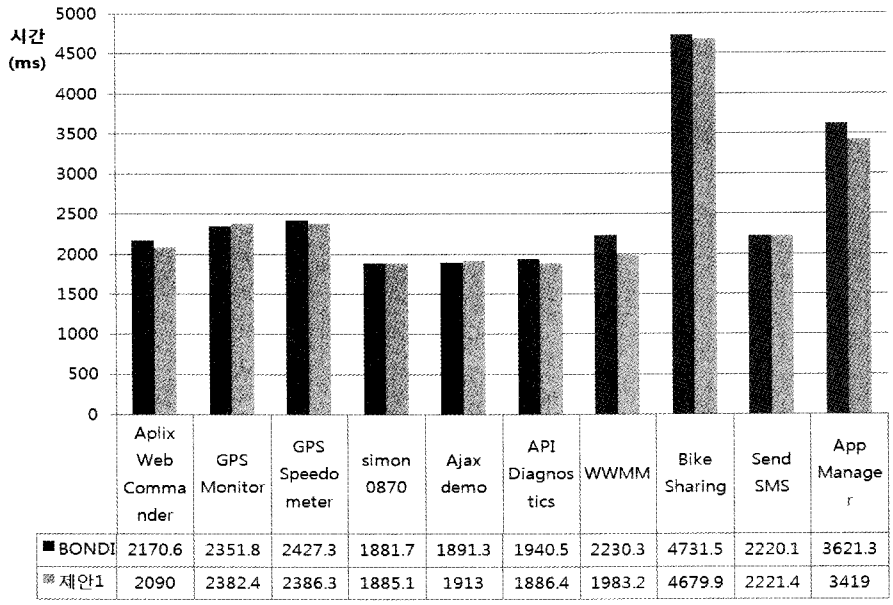
4.2.2 Widget Resource List 재 정의를 통한 성능 평가

모듈별 Widget Configuration Document 요소 추가 후 실험 결과를 통하여 기존 BONDI 에 규정된 모듈 수에 따른 속도 차이를 확인 할 수 있었다. 추가



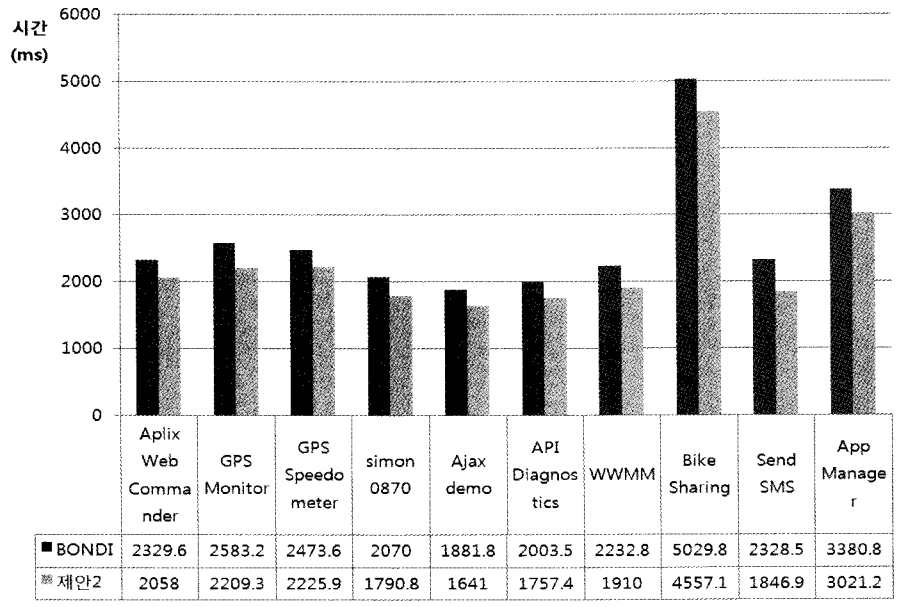
	Aplix Web Commander	GPS Monitor	GPS Speedometer	simon 0870	Ajax demo	API Diagnostics	WWMM	Bike Sharing	Send SMS	App Manager
BONDI	37.5	45.5	47.2	43.2	43.6	22.4	43.5	35.3	38	40.4
제안1	24.2	7.4	8.8	8.1	4.5	5.5	10.9	8.8	17.6	11.3

그림 9. BONDI 위젯의 모듈 실행 속도 성능 평가



	Aplix Web Commander	GPS Monitor	GPS Speedometer	simon 0870	Ajax demo	API Diagnostics	WWMM	Bike Sharing	Send SMS	App Manager
BONDI	2170.6	2351.8	2427.3	1881.7	1891.3	1940.5	2230.3	4731.5	2220.1	3621.3
제안1	2090	2382.4	2386.3	1885.1	1913	1886.4	1983.2	4679.9	2221.4	3419

그림 10. BONDI 위젯 Full-screen 속도 성능 평가



	Aplix Web Commander	GPS Monitor	GPS Speedometer	simon 0870	Ajax demo	API Diagnostics	WWMM	Bike Sharing	Send SMS	App Manager
BONDI	2329.6	2583.2	2473.6	2070	1881.8	2003.5	2232.8	5029.8	2328.5	3380.8
제안2	2058	2209.3	2225.9	1790.8	1641	1757.4	1910	4557.1	1846.9	3021.2

그림 11. WRList 재 정의를 통한 BONDI 위젯 실행 성능 평가

적으로 Widget Resource List에 위젯이 설치될 때의 Configuration Document 정보를 사전에 읽고 저장하여 설치 및 로딩의 절차를 간소화한 기법을 이용하여 성능 평가를 실시하였고 그 결과가 그림 11에 보이고 있다.

BONDI의 Widget Resource List는 위젯 리소스의 URL 정보만 가져올 수 있기 때문에 Widget User Agent가 위젯 설치 및 로딩 시 엔진에서 Widget ID, Widget Version, Widget icon 등의 정보를 얻을 때 항상 위젯 리소스의 'config.xml'을 참조해야 했다. 본 연구에서 고안된 'WRList 재 정의를 통한 기법'을 사용하여 그림 11에서 측정값을 분석한 결과, 선별되

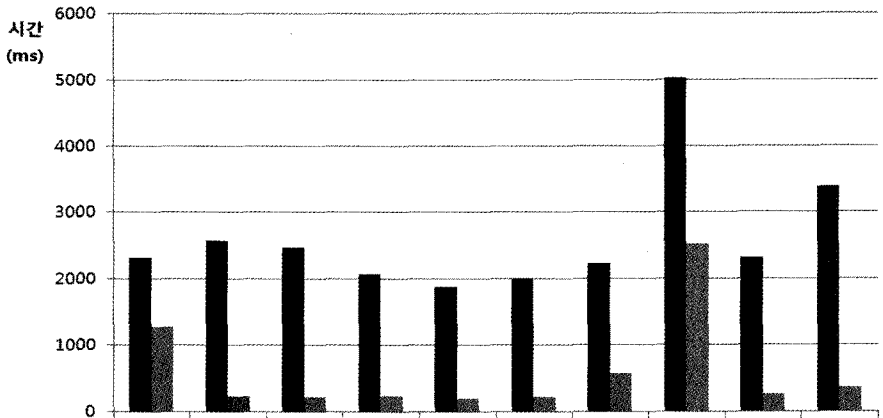
어 실험에 사용된 열 개의 위젯 모두 200*ms*에서 500*ms* 정도 더 빠르게 실행되는 것을 확인하였다. 400ms가 넘는 차이를 보이는 위젯은 Bike Sharing과 Send SMS로 해당 위젯의 이미지 및 CSS, JavaScript 파일 용량이 크기 때문이며, 측정 시 네트워크 사용 등 단말기의 상태에 따라 미세한 값의 변화가 있을 수 도 있다.

4.2.3 Virtual Widget Cache 사용을 통한 위젯 실행 시간 성능 평가

기존 BONDI는 위젯 설치 및 로딩 시 Widget Resource List의 URL을 참조하여 Configuration Document의 정보를 읽도록 하였는데 본 연구에서는 자주 사용하는 위젯의 빠른 실행을 위해 Virtual Widget Cache를 이용하여 Widget Resource List를 보다 효율적으로 사용하도록 설계 하였다. Configuration Document 정보를 읽기 위해서는 압축된 위젯 리소스를 풀어야 하는데 실행 후 종료 시 압축을 푼 위젯 리소스를 삭제하도록 한 것 때문에 BONDI에 지정된 방법으로 위젯 정보를 저장하면 재차 사용 시 효율적인 운용이 어렵다. 그림 12는 Virtual Widget Cache를 이용한 개선 기법을 활용한 성능 평가 결과이다.

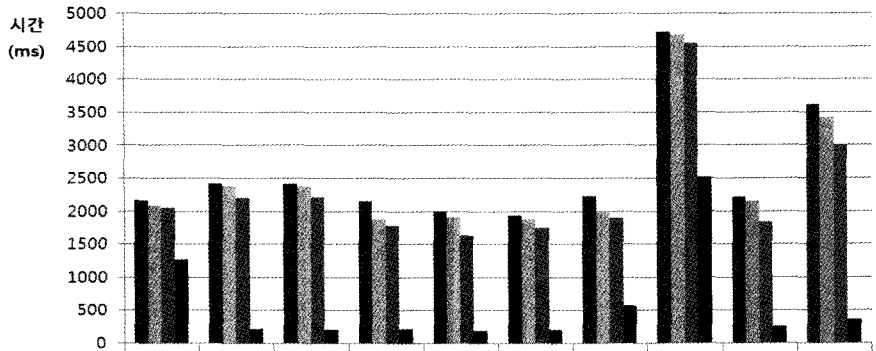
그림 12에서는 열 개의 위젯 모두 기존 BONDI와 Widget Resource List 재 정의 기법을 이용한 성능 평가 결과보다 더 향상된 내용을 보이고 있다. 마지막으로 모듈별 Configuration Document 요소 추가 기법 (제안 1), Widget Resource List 재 정의 기법 (제안 2), Virtual Widget Cache 사용 기법 (제안 3)들을 종합적으로 조합하여 성능을 평가하고 분석하여 최종 결과를 도출하였다. 결과 비교를 위한 대조군은 기존의 전체모듈을 로드하는 BONDI에 규정된 방법이고 실험군으로는 첫째, 위젯이 필요로 하는 모듈만 로드하는 Configuration Document 요소 추가 기법, 둘째, 첫째 방법에 Widget Resource List 재 정의 기법을 추가한 방법, 셋째, 둘째 방법에 Virtual Widget Cache 사용 기법을 추가한 방법으로 하였다. 그림 13은 이러한 기법들의 조합을 통해 도출된 최종 결과를 나타내고 있다. 이 그림을 통하여 본 연구에서 제시한 방법을 조합하여 다수의 기법으로 실험한 결과, 위젯의 실행 속도가 대폭 개선되었음을 알 수 있다. 특히 Virtual Widget Cache를 사용하는 기법 (제안 3)을 추가하였을 시는 확연히 눈에 띄는 차이를 보이고 있다.

그림 13에서 각각의 위젯 용량이나 사용 모듈에 따라 차이가 있지만 열 개의 위젯 모두 실행 속도가 대폭 개선되었으며 실제 상용화 단말인 Motorola의 XT720에서 테스트 할 때에도 BONDI에서 규정한 방



	Aplix Web Commander	GPS Monitor	GPS Speedometer	simon 0870	Ajax demo	API Diagnostics	WWMM	Bike Sharing	Send SMS	App Manager
■BONDI	2329.6	2583.2	2473.6	2070	1881.8	2003.5	2232.8	5029.8	2328.5	3380.8
※제안3	1270.5	230	211	230.4	196.8	219.4	576.8	2524.7	259.7	361.1

그림 12. Virtual Widget Cache를 이용한 위젯 실행 성능평가



	Aplix Web Commander	GPS Monitor	GPS Speedometer	simon 0870	Ajax demo	API Diagnostics	WWMM	Bike Sharing	Send SMS	App Manager
BONDI	2170.6	2427.6	2427.3	2163.2	2007.7	1940.5	2230.3	4731.5	2220.1	3621.3
제안1	2090	2382.4	2386.3	1885.1	1913	1886.4	1983.2	4679.9	2154.7	3419
제안1+2	2058	2209.3	2225.9	1790.8	1641	1757.4	1910	4557.1	1846.9	3021.2
제안1+2+3	1270.5	230	211	230.4	196.8	219.4	576.8	2524.7	259.7	361.1

그림 13. 세 가지 기법 조합을 통한 위젯 실행 속도 성능 평가

법 보다 더 향상된 결과를 보였다. 향 후 cache를 사용하면서 해당 모바일 단말기의 메모리를 많이 사용할 수 있어 이를 조절하기 위한 효율적인 cache 운용 알고리즘 연구가 필요할 것으로 사료되며 이를 통해 위젯 실행 속도 뿐 만 아니라 메모리 사용량을 줄이며 안정성과 구동 시의 신뢰성을 동시에 높이는 방법을 연구해야 할 것으로 보인다.

5. 결론 및 향후 연구

본 논문에서는 OMTP BONDI 기반 모바일 웹 위젯 리소스의 효율적인 운용 및 관리를 위한 구동 성능 개선 기법을 연구하고 그 결과를 제시, 분석하였다. OMTP BONDI 위젯은 모바일 단말기의 하드웨어 및 소프트웨어 자원을 사용하기 위해 지정된 모듈을 사용하고 있다. BONDI 위젯은 사용하고자 하는 모듈을 <feature>요소를 통해 정의 할 수는 있지만 위젯 엔진 자체에서 사용하지 않는 BONDI 모듈도 동시에 로드한다. 이는 악의적인 사용자에 의해 만들어진 위젯을 다운로드하여 설치, 실행하였을 시 사용자의 개인 정보 누출과 같은 보안 위험이 존재할 수 있고, 위젯의 실행속도 또한 느려지게 된다. 본 논문에서는 위젯 실행 시 BONDI 모듈을 선택적으로 로드하여 모듈 실행 속도를 개선하는 기법과, 표준화가 진행 중 인 BONDI 위젯의 효율적 운용 및 관리 방안을 제시하였으며 기법 조합을 통해 성능 평가를 진행하고 실험결과를 비교, 분석 하였다. 첫째, Widget Configuration Document에 구성 요소를 추가하는 기법, 둘째, Widget Resource List를 재 정의하고 항목을 추가하는 기법, 셋째, Virtual Widget Cache를 고안하여 활용하는 기법을 이용하여 성능 평가를 진행하였고 그 결과 본 연구에서 제시한 기법들이 실행도 면에서 향상된 결과를 도출하였음을 확인할 수 있었다. 최종적으로는 위에 제시한 첫 번째, 두 번째 그리고 세 번째 방법을 서로 조합하여 성능 평가를 진행하였고 그 결과 수행도면에서 만족할 만한 성과를 보였다. 또한 Widget BOX를 고안하여 사용하던 위젯을 중요 정보와 함께 원격 서버에 업로드하여 저장하는 방식을 채택하였고 다시 사용하고자 할 때 언제든지 다운로드 받을 수 있게 하였다. 향 후 무선 단말의 메모리 사용량을 조절하기 위한 적응적 cache 운용 알고리즘 연구가 필요할 것으로 보이며 동시에 위젯 실행 속도를 더욱 향상 시키고 리소스 사용량을 줄여 안정성과 신뢰성을 함께 개선하는 방법을 연구할 계획이다.

참 고 문 헌

[1] 조성선, "무선인터넷 접속 규격-WAP," http://kidbs.itfind.or.kr/WZIN/jugidong/920/

92004.html
[2] 이원석, "HTML5와 모바일 웹," 한국전자통신연구원, 2010.
[3] "HTML 5," W3C Working Draft, http://www.w3.org/TR/html5/, 2010.
[4] "HTML 5 Differences form HTML 4," W3C Working Draft, http://www.w3.org/TR/2009/WD- html5-diff-20090423/, 2009.
[5] "BONDI Architecture and Security Application Lifecycle v1.0," http://bondi.omtp.org/1.1/security/BONDI_Architecture_and_Security_Application_Lifecycle_v1.0.pdf, 2009.
[6] "JIL Widget Engine Overview White Paper," http://jil.org/, 2009.
[7] 권기덕, "스마트폰이 IT 시장에 미치는 영향 SW Insight," 정책리포트, 2009.
[8] A. Taivalsaari, T. Mikkonen, D. Ingalls, and K. Palacz, "Web Browser as an Application Platform," SEAA '08. 34th Euromicro Conference Software Engineering and Advanced Applications, pp. 293-302, 2008.
[9] "Feature-phone vs Smart-phone," http://mobizen.pe.kr/891, 2010.
[10] 류중희, "Web 2.0에서 Mobile 2.0으로," http://ignoblesse.tistory.com/tag/mobile%202.0, 2010.
[11] "W3C Mobile Web 2.0 Forum," http://www.mw2.or.kr, 2010.
[12] F. Reynolds, "Web 2.0-In Your Hand," *Pervasive Computing, IEEE, Jan.-March*, pp. 86-88, 2009.
[13] T. O'Reilly, "What is Web 2.0?," http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html, 2010.
[14] "BONDI 1.1 Approved Release," http://bondi.omtp.org/1.1/, 2009.
[15] "BONDI 1.5 APIs Public Working Draft v1," http://bondi.omtp.org/1.5/pwd-1/, 2009.
[16] "BONDI API Specification - Version 1.1," http://bondi.omtp.org/1.1/, 2010.
[17] "The BONDI API Design Patterns - Version 1.1," http://bondi.omtp.org/1.1/apis/BONDI_Interface_Patterns_v1.1.html, 2009.
[18] 이성준, 김대원, "PoC Box 시스템이 적용된 모바일 환경에서 단말로의 효율적인 전송을 위한 RTSP 기반 미디어 표현 및 구조 생성 방법," 한국멀티미디어학회논문지, Vol. 12, No. 8, pp. 1142-1154, 2009.

방 지 웅

2008년 2월 단국대학교 컴퓨터과학과 공학사
2010년 2월 단국대학교 대학원 컴퓨터과학과 공학석사
2010년 3월~현재 단국대학교 대학원 컴퓨터과학과 박사과정
YAMAIA-NGTMS 기술연구소 연구원
관심분야: 모바일 응용, 임베디드 시스템, 인공지능

김 대 원

1993년 2월 중앙대학교 전자공학과 공학사
1996년 5월 USC(So. Cal.)-EE 공학석사
2002년 5월 ISU(Iowa St.)-EE 공학박사
2002년 5월~2004년 8월 삼성전자 TN총괄 통신연구소, 차세대 단말기술 Lab, 책임연구원
2004년 9월~현재 단국대학교천안캠퍼스 공학대학 멀티미디어공학과 부교수
관심분야 : 멀티미디어 신호처리, 모바일 응용, NDE